

REMARKS

Reconsideration of the application in light of the following remarks is respectfully requested.

I. Status of the Claims

Claims 1-3 are pending and are presented as a courtesy to the Examiner.

II. Inventorship

Applicants thank the Examiner for acknowledging the change of inventorship. Accordingly, the Declaration under 37 C.F.R. § 1.131 (see below) has been executed by all three inventors.

III. Rejections Under 35 U.S.C. § 102

Claims 1-3 are rejected under 35 U.S.C. § 102(e) as anticipated by U.S. Patent Publication No. 2004/0139483 to Kim et al. ("Kim"). Applicants respectfully traverse the rejection.

Applicants disagree that Kim, or any of its related references, shows all of the features of the claimed invention. Nevertheless, Applicants respectfully traverse the rejection by directing the Examiner to the Declaration under 37 C.F.R. § 1.131 (along with documentary evidence in the form of Exhibits B and C) filed concurrently herewith. The Declaration is signed by all of the Applicants and states that the Applicants had completed the invention as claimed in the present application prior to the earliest filing date of Kim.

Exhibits B & C of the Declaration describe the architecture for the method and system of Applicants' claims and describe the claimed features in detail. In addressing the Examiner's comments, Applicants submit that all of the elements are supported by the code disclosed in Exhibit B. Applicants have excerpted below, in more detail, the supporting code for the elements

questioned by the Examiner. The code in Exhibit B is identical to the code provided in support of the previous Declaration and Amendment. Applicants submit that the subject matter is correlated to the text of the claims.

With respect to claims 1 and 3, Exhibit B discloses numerous elements of the claims. The subroutine on pages 1-5 discloses the identification system that identifies the plurality of content files. Excerpts from the subroutine illustrate the element:

```
sub InputIDData {  
  my %stepDesc = @_;  
  my $pageTitle = "UMG Multimedia Pre-Mastering Database: Input ID Data";  
  my $instructions = "You must fill out all fields with a yellow background. All other fields (blue \\  
  background) are optional. Please enter identifying data about this ECD.";  
  ...  
  # display form for ECD data input# print the album data input table# prompt for multimedia output type (ECD,  
  DataPlay, or both) ...  
  # in order to save state, print all of the entry data that isn't ...  
  # editable as hidden fields ...  
  # sub InitInputTableData - initialize the descriptions of all input tables ...  
  # now, add the dynamic tables based on user input data ...  
  # the platform independent table data# now, initialize the list of fields specified for each step ...  
  # the fields for id data tables# the fields for content description tables.
```

Another subroutine discloses the relation system relating the related content files. Excerpts from pages 6 and 7 of Exhibit B disclose:

```
# sub SaveEntryData - Save the field data values to a text file ...  
# selection number must be defined since it's used as the  
# hash for each ecd's data ...  
# copy the field data values into a hash of their own ...  
sub WriteUMGMMData - write data for selected disc  
#       current implementation reads all data, adds/modifies data ...  
#       for selected disc, then writes all data  
# passed: umgmm data hash to be written ...  
# read %allUMGMMs from file ...  
#       PrintAllUMGMMData (%allUMGMMData);  
# copy this disc's data to the allUMGMM hash  
# for now, use the selecton number as the hash key ...  
# write the new file.
```

Further, a query system querying a server for content files is described in Exhibit B, pages 7-9.

Particularly, on page 7, this code moves and extracts files from one location on the server to another location:

```
# extract the ecd template files
print "Extracting DataPlay template archive ...", br;
ExtractTemplateFiles ($starRootPath, $dptemplate);
```

Also, Exhibit B, page 9 discloses a subroutine that extracts files and directories from an archive and saves them to the serving server:

```
sub ExtractTemplateFiles {
    my $starRootPath = shift @_;
    my $tartemplate = shift @_;

    # chdir to the destination dir
    chdir($starRootPath) or die "Can't chdir $starRootPath: $!";

    # read the source tar file
    print "about to read $tartemplate", br;
    my $tar = new Archive::Tar($tartemplate);

    # write the source tar files to the new directory
    my @files = $tar->list_files;
    $tar->extract(@files);
}
```

Storing information about the plurality of content files in a reference database is disclosed in the subroutine on pages 9-12 of Exhibit B that discloses the “# umgmmDBManager.pm - module for managing reading & writing data to the umgmultimaster DB [Note: DB is shorthand for Database].”

A collection system that allows a user to create the plurality of content files into at least one collection file is disclosed at pages 12-13 of Exhibit B:

```
sub CreateDataPlayOutput {

    # untar the dataplay template directories
    my $starRootPath = catfile ($fileoutputdir, $USER_INPUT_DATA{'cdname'}{'value'});

    # extract the ecd template files
    print "Extracting DataPlay template archive ...", br;
    ExtractTemplateFiles ($starRootPath, $dptemplate);
```

```
# rename the extracted directory tree to the name of this cd
chdir($starRootPath) or die "Can't chdir $starRootPath: $!";
my $dpDir = $USER_INPUT_DATA{cdname}{value}."DP";
rename "dpFileTemplate", $dpDir;

# write Contents.ddl to root/"Content Manager" directory
my $contentDir = catfile ($starRootPath, $dpDir, "Content Manager");
if (! -e ($contentDir)) {
    mkdir $contentDir, 0777 or die "Can't mkdir $contentDir: $!";
}

my $contentsDDLTemplate = catfile ($fileinputdir, "Contents.ddl.tpl");
my $outputFileName = catfile ($contentDir, "Contents.ddl");
FillAndWriteTemplateFile ($contentsDDLTemplate, $outputFileName);

# populate root/Music/_Playlists_
my $musicDir = catfile ($starRootPath, $dpDir, "Music");
if (! -e ($musicDir)) {
    mkdir $musicDir, 0777 or die "Can't mkdir $musicDir: $!";
}
my $playlistDir = catfile ($musicDir, "_Playlists_");
if (! -e ($playlistDir)) {
    mkdir $playlistDir, 0777 or die "Can't mkdir $playlistDir: $!";
}
chdir($playlistDir) or die "Can't chdir $playlistDir: $!";

my $defaultPlaylistTemplate = catfile ($fileinputdir, "Default.playlist.tpl");
my $outputFileName = catfile ($playlistDir, "Default.playlist");
FillAndWriteTemplateFile ($defaultPlaylistTemplate, $outputFileName);

# create the album directory under root/Music
chdir($musicDir) or die "Can't chdir $musicDir: $!";
my $albumDir = catfile ($musicDir, $USER_INPUT_DATA{albumname}{value});
if (! -e ($albumDir)) {
    mkdir $albumDir, 0777 or die "Can't mkdir $albumDir: $!";
}

# populate the album directory with "Thumbnail.jpg" and album-
# specific files (video, audio, text, images, credits, etc.)

# create the track directories under root/Music/Album Title
chdir($albumDir) or die "Can't chdir $albumDir: $!";
for (my $i=0; $i<$USER_INPUT_DATA{nsongs}{value}; $i++) {
    my $songKey = "songname$i";
    my $songDir = $USER_INPUT_DATA{$songKey}{value};
    if (! -e ($songDir)) {
        mkdir $songDir, 0777 or die "Can't mkdir $songDir: $!";
    }
}
```

```
# populate the track directories with "Thumbnail.jpg" for each track
# populate the track directories with track-specific files (lyrics, credits, etc.)

print "Writing compressed DataPlay output file ...", br;
WriteTarFile ($starRootPath, $dpDir);

print a ({-href=>"$ftpdDir/$dpDir/$dpDir.tar"},
  "Click Here to download the compressed DataPlay file for:
  $USER_INPUT_DATA{albumname}{value}"), br;
```

This subroutine creates and populates the file structure for a disk, compiles the file structure into a single file and displays a link for a user to download the file.

Furthermore, the code listed below (*see*, Exhibit B, page 15), extracts a file structure from a single file, creates and saves a control file defining the content on the disk, then creates a single file with the file structure and control file. A link is then displayed for the user to download the file.

```
sub CreateECDOutput {

  my $starRootPath = catfile ($fileoutdir, $USER_INPUT_DATA{cdname}{value});

  # extract the ecd template files
  print "Extracting ECD template archive ...", br;
  ExtractTemplateFiles ($starRootPath, $ecdtartemplate);

  # rename the extracted directory tree to the name of this cd
  chdir($starRootPath) or die "Can't chdir $starRootPath: $!";
  my $ecddir = $USER_INPUT_DATA{cdname}{value}."ECD";
  rename "ecdFileTemplate", $ecddir;

  print "Creating Script files ...", br;
  CreateDCDLFile ($starRootPath, $ecddir);

  print "Writing compressed ECD output file ...", br;
  WriteTarFile ($starRootPath, $ecddir);

  print a ({-href=>"$ftpdDir/$ecddir/$ecddir.tar"},
    "Click Here to download the compressed ECD file for:
    $USER_INPUT_DATA{albumname}{value}"), br;
```

Additionally, a conversion module to transfer to an authoring system to convert at least one collection file into a master copy, wherein the master copy is in one of a plurality of formats is embodied in the subroutine on pages 14-15 of Exhibit B:

```
sub OutputFiles - Create the appropriate out files, including:
#           - untarring the template directory structure (for ECD's)
#           - creating the dcdl file (for ECD's)
#           - untarring the DataPlay dir structure
#           - creating relevent DataPlay files ...
my $pageTitle = "UMG Multimedia Pre-Mastering Database: Output Files";
my $instructions = "The compressed ECD file has been created. It is in zipped tar
format. To download the file, click on the link below."; ...
# extract the ecd template files ...
# rename the extracted directory tree to the name of this cd ..
print "Writing compressed ECD output file ...", br; "Click Here to download the compressed ECD file for: ...
    $USER_INPUT_DATA{albumname}{value}"), br.
```

The URL configuration system identifying a URL and communicating with a redirect server to activate the URL is disclosed in Exhibit C. The conception of the elements are embodied in the commentary between the inventors, senior management and the programmers. Quotes discussing the system include “With new UMG portal, do we want to actually house the site at the /umusic directory on the server, and then redirect any hard coded references to “www.universalstudios.com/music” to the “www.umusic.com” site?” (Exhibit C1). A “redirect URL ... will allow us to do two things: 1. Fix broken, dead, or hostile links in the future[; and] 2. Track DVD-A users who visit our URL redirect for the compilation of consumer data.” (Exhibit C2). “[W]e had talked about setting up a redirect at universal so that it would always point to the most up to date QT location.” (Exhibit C3). “Create redirects/web pages for: ...” (Exhibit C4). Also, “We need to implement the URL redirect stuff.” (Exhibit C5).

Thus, independent claims 1 and 3 are supported by the disclosure of Exhibits B and C. Further, the Declaration states that the invention was made prior to February 21, 2002. The

Exhibits, as outlined above, clearly show that the invention was fully completed before the effective date of the reference and tested to verify that the source code ran successfully.

Also, claim 2 depends from claim 1 and thus contain all of its limitations per 35 U.S.C. § 112, fourth paragraph. Additionally, MPEP § 2131 states that a “claim is anticipated only if each and every element as set forth in the claim is found... in a single prior art reference.” Thus, if the elements of claim 1 are antedated, the reference is not prior art, and all of the claims have elements not in the prior art, all of the independent and dependent claims are allowable.

Thus, Applicants respectfully request that the rejection be withdrawn.

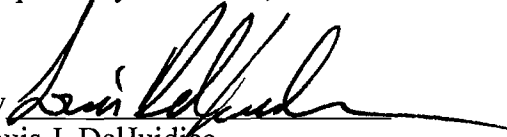
CONCLUSION

In view of the above amendments, Applicants believe the pending application is in condition for allowance. In view of the above, each of the presently pending claims in this application is believed to be in immediate condition for allowance. Accordingly, the Examiner is respectfully requested to pass this application to issue.

The Examiner is respectfully requested to contact the undersigned at the telephone number indicated below once he has reviewed the proposed amendment if the Examiner believes any issue can be resolved through either a Supplemental Response or an Examiner's Amendment.

Dated: July 23, 2007

Respectfully submitted,

By 

Louis J. DelJuidice

Registration No.: 47,522

DARBY & DARBY P.C.

P.O. Box 5257

New York, New York 10150-5257

(212) 527-7700

(212) 527-7701 (Fax)

Attorneys/Agents For Applicant